US009191327B2

(12) United States Patent
Shieh

(10) **Patent No.: US 9,191,327 B2**

(45) **Date of Patent: *Nov. 17, 2015**

(54) **DISTRIBUTED SERVICE PROCESSING OF NETWORK GATEWAYS USING VIRTUAL MACHINES**

(75) Inventor: **Choung-Yaw Michael Shieh**, Palo Alto, CA (US)

(73) Assignee: **VARMOUR NETWORKS, INC.**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 832 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/363,082**

(22) Filed: **Jan. 31, 2012**

(65) **Prior Publication Data**

US 2012/0207174 A1 Aug. 16, 2012

Related U.S. Application Data

(60) Provisional application No. 61/462,980, filed on Feb. 10, 2011.

(51) **Int. Cl.**

H04L 12/803 (2013.01)
H04L 29/06 (2006.01)
H04L 12/813 (2013.01)

(52) **U.S. Cl.**

CPC ***H04L 47/125*** (2013.01); ***H04L 63/0209*** (2013.01); *H04L 47/20* (2013.01)

(58) **Field of Classification Search**

None

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,765,864	B1	7/2004	Natarajan et al.	
6,992,985	B1	1/2006	Das	
7,058,712	B1 *	6/2006	Vasko et al.	709/224
7,062,566	B2	6/2006	Amara et al.	
7,774,837	B2	8/2010	McAlister et al.	
7,849,495	B1	12/2010	Huang et al.	
8,274,912	B2	9/2012	Wray et al.	
8,321,862	B2	11/2012	Swamy et al.	
2003/0177389	A1	9/2003	Albert et al.	
2004/0095897	A1	5/2004	Vafaei	
2007/0079308	A1	4/2007	Chiaramonte et al.	

(Continued)

FOREIGN PATENT DOCUMENTS

WO WO 2011/012165 2/2011

OTHER PUBLICATIONS

Office Action for related U.S. Appl. No. 13/288,872, Jun. 24, 2013, 10 pages.

(Continued)

Primary Examiner — Christine Duong

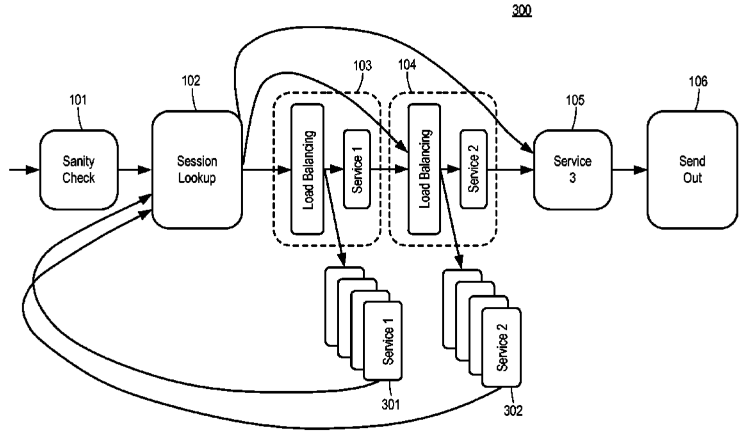
(74) *Attorney, Agent, or Firm* — Blakely, Sokoloff, Taylor & Zafman LLP

(57) **ABSTRACT**

A network gateway device includes an ingress interface, an egress interface, and a load balancing module coupled to the ingress and egress interfaces. The load balancing module configured to receive a packet from the ingress interface, determine a set of a plurality of processes corresponding a connections session associated with the packet based on a policy. For each of the identified processes, the load balancing module is to identify a service processing module executed by a virtual machine that is capable of handling the identified process, and to send the packet to the identified service processing module to perform the identified process on the packet. The packet is then transmitted to the egress interface of the gateway device to be forwarded to a destination.

18 Claims, 8 Drawing Sheets

300



(56) **References Cited**

U.S. PATENT DOCUMENTS

2007/0271612	A1 *	11/2007	Fang et al.	726/22
2008/0083011	A1	4/2008	McAlister et al.	
2008/0155239	A1	6/2008	Chowdhury et al.	
2008/0301770	A1	12/2008	Kinder	
2010/0043068	A1	2/2010	Varadhan et al.	
2010/0095367	A1	4/2010	Narayanaswamy	
2010/0228962	A1 *	9/2010	Simon et al.	713/150
2010/0235880	A1	9/2010	Chen et al.	
2011/0003580	A1	1/2011	Belrose et al.	
2011/0069710	A1	3/2011	Naven et al.	
2011/0138384	A1 *	6/2011	Bozek et al.	718/1
2011/0225624	A1	9/2011	Sawhney et al.	
2011/0263238	A1	10/2011	Riley et al.	
2012/0017258	A1	1/2012	Suzuki	
2012/0131685	A1	5/2012	Broch et al.	
2012/0311144	A1 *	12/2012	Akelbein et al.	709/224
2012/0311575	A1	12/2012	Song	
2013/0086399	A1	4/2013	Tychon et al.	
2013/0097692	A1	4/2013	Cooper et al.	

OTHER PUBLICATIONS

Office Action for related U.S. Appl. No. 13/288,872, Jan. 22, 2013, 9 pages.

Office Action for related U.S. Appl. No. 13/363,088, Feb. 13, 2013, 6 pages.

Notice of Allowance for related U.S. Appl. No. 13/363,088, Aug. 16, 2013, 4 pages.

Office Action for related U.S. Appl. No. 13.285,814, Dec. 11, 2012, 20 pages.

Office Action for related U.S. Appl. No. 13.285,814, Aug. 20, 2013, 23 pages.

U.S. Appl. No. 13/288,872, Notice of Allowance and Fee(s) Due, Dated Apr. 11, 2014, 19 pages.

U.S. Appl. No. 13/285,814, Office Action, Mar. 26, 2014, 26 pages.

U.S. Appl. No. 13/291,739, Office Action, May 8, 2013, 13 pages.

U.S. Appl. No. 13/291,739, Final Office Action, Oct. 16, 2013, 19 pages.

U.S. Appl. No. 13/291,739, Office Action, Oct. 21, 2014, 14 pages.

U.S. Appl. No. 13/291,739, Office Action, May 19, 2015, 12 pages.

* cited by examiner

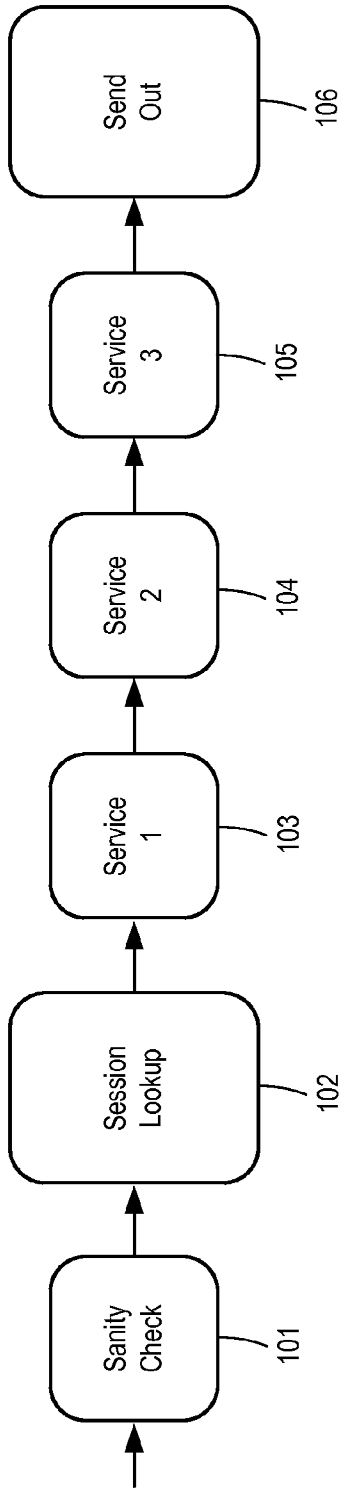


FIG. 1
(Prior Art)

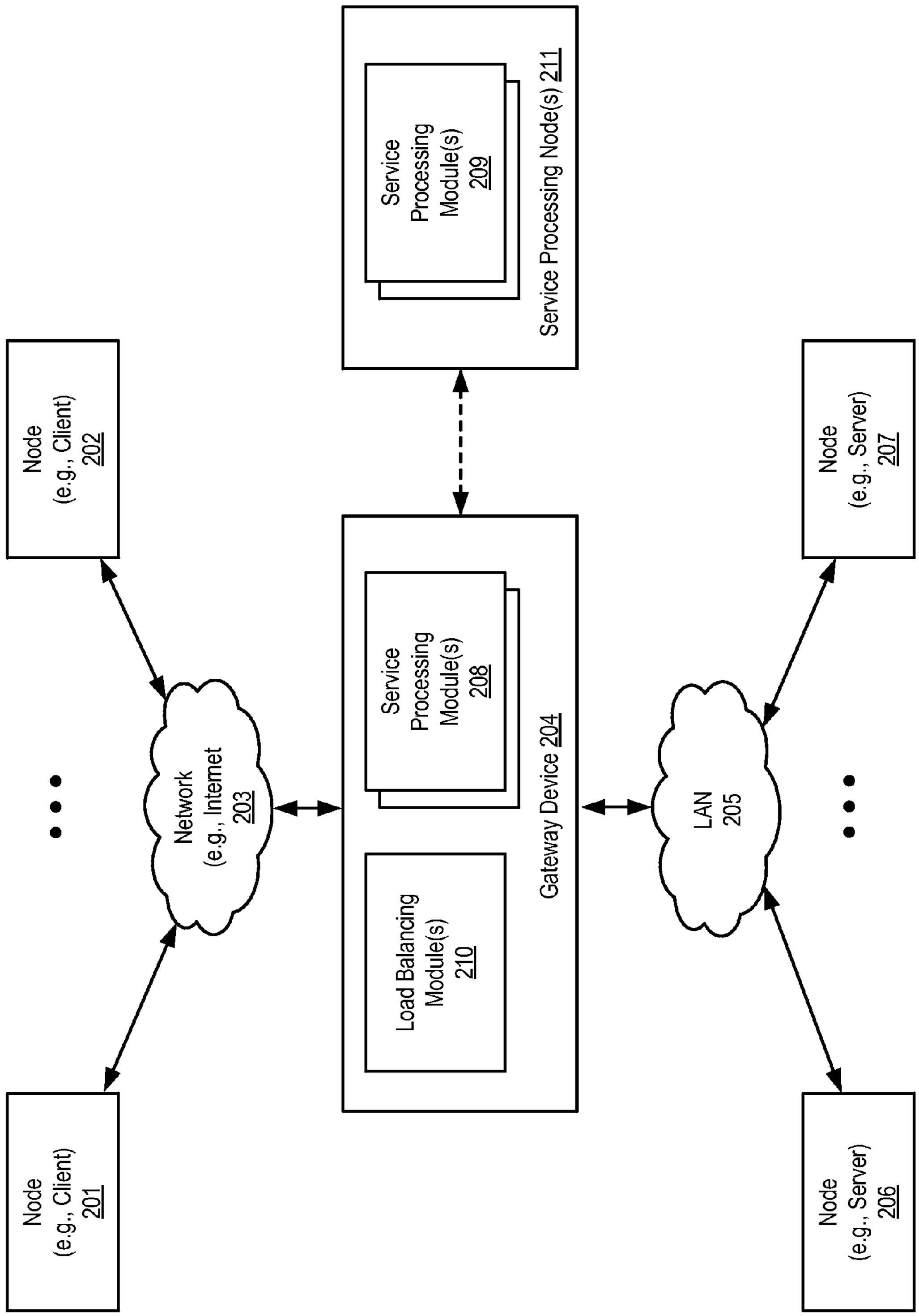

Service Processing Module(s) 209
Service Processing Node(s) 211
Node (e.g., Client) 202
Node (e.g., Server) 207
Network (e.g., Internet) 203
Service Processing Module(s) 208
Load Balancing Module(s) 210
Gateway Device 204
LAN 205
Node (e.g., Client) 201
Node (e.g., Server) 206

FIG. 2

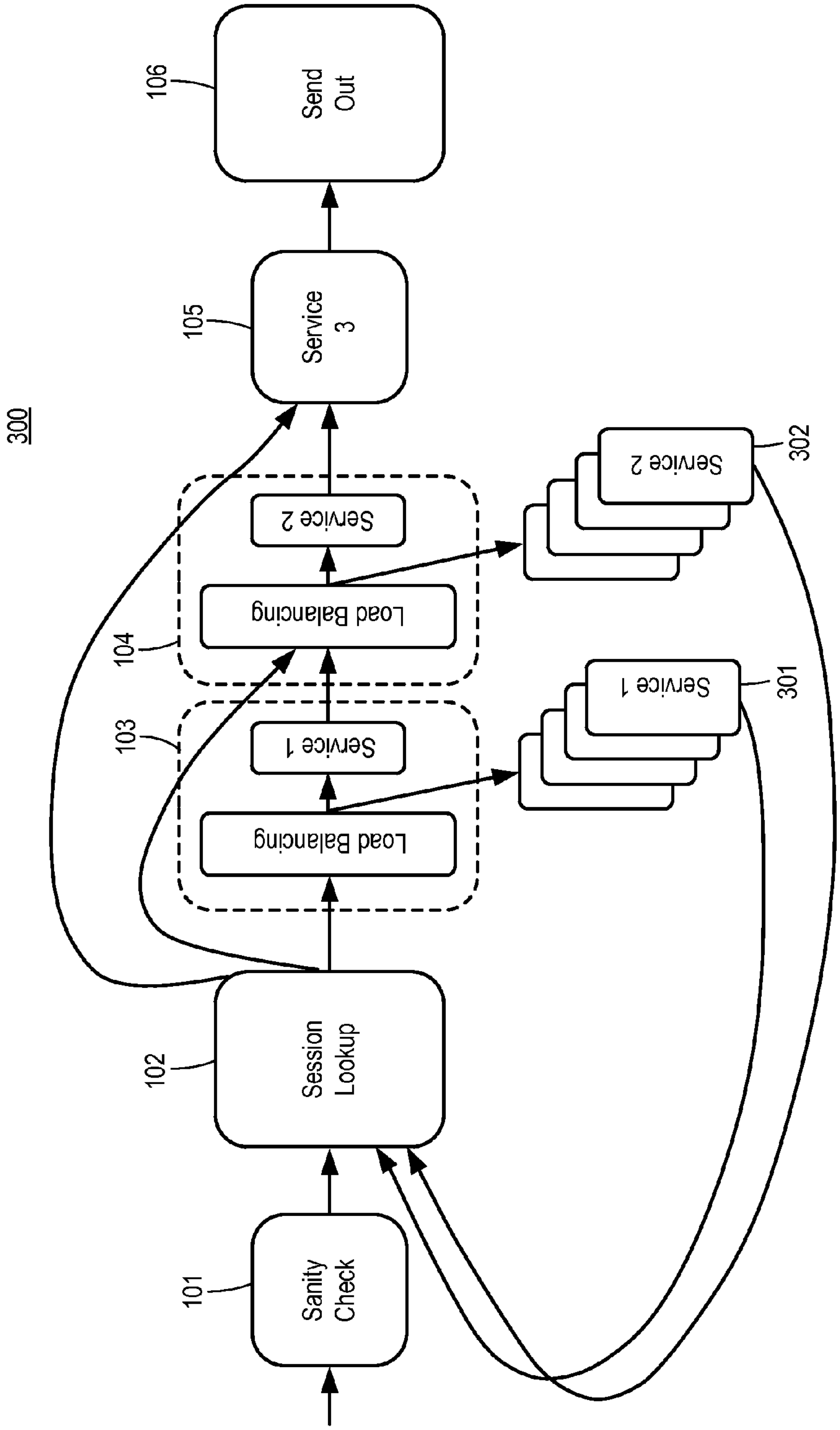

300
101
Sanity Check
102
Session Lookup
103
Load Balancing
Service 1
104
Load Balancing
Service 2
105
Service 3
106
Send Out
301
Service 1
302
Service 2

FIG. 3

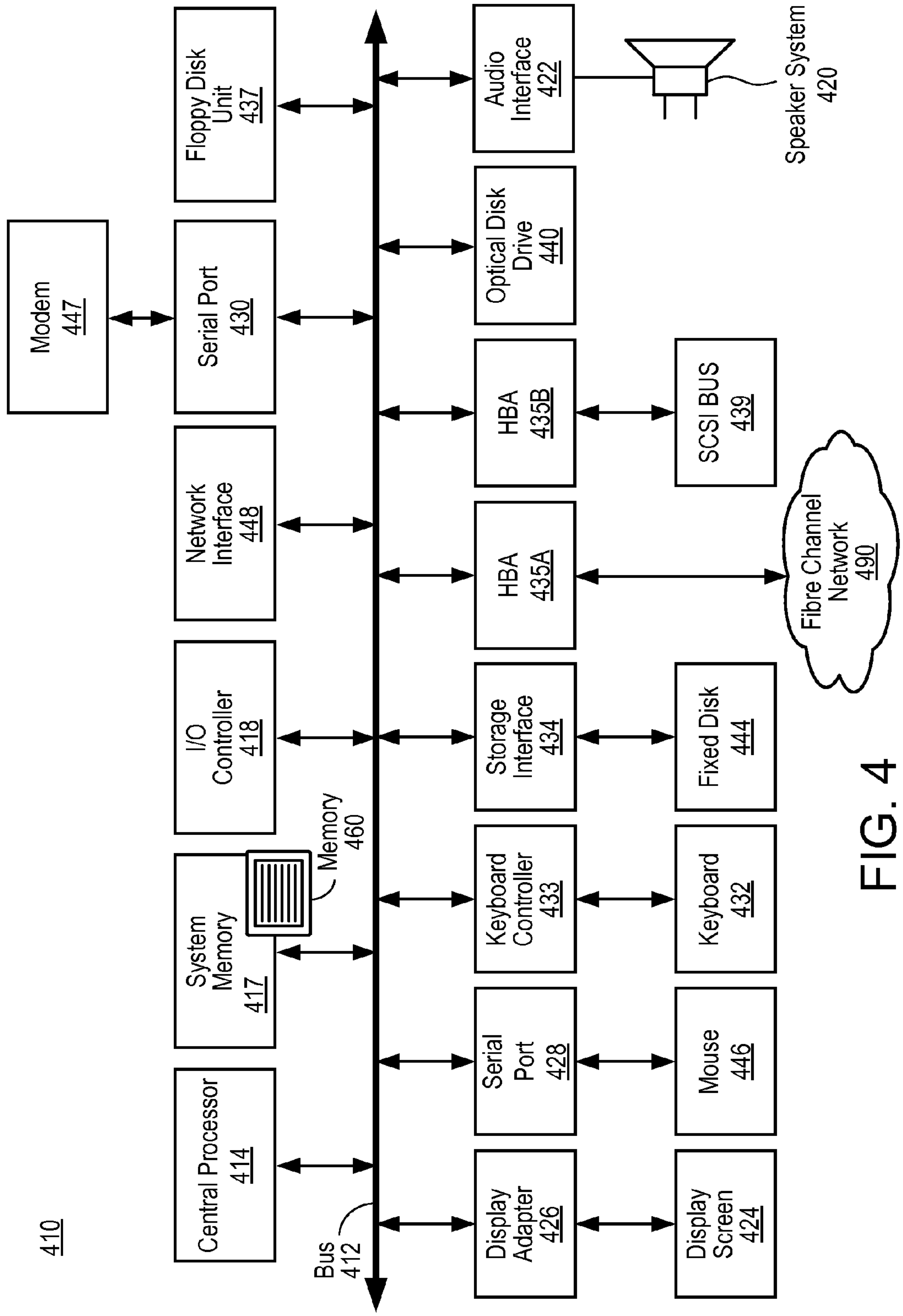

410
Central Processor 414
System Memory 417
Memory 460
I/O Controller 418
Network Interface 448
Serial Port 430
Modem 447
Floppy Disk Unit 437
Bus 412
Display Adapter 426
Serial Port 428
Keyboard Controller 433
Storage Interface 434
HBA 435A
HBA 435B
Optical Disk Drive 440
Audio Interface 422
Speaker System 420
Display Screen 424
Mouse 446
Keyboard 432
Fixed Disk 444
Fibre Channel Network 490
SCSI BUS 439

FIG. 4

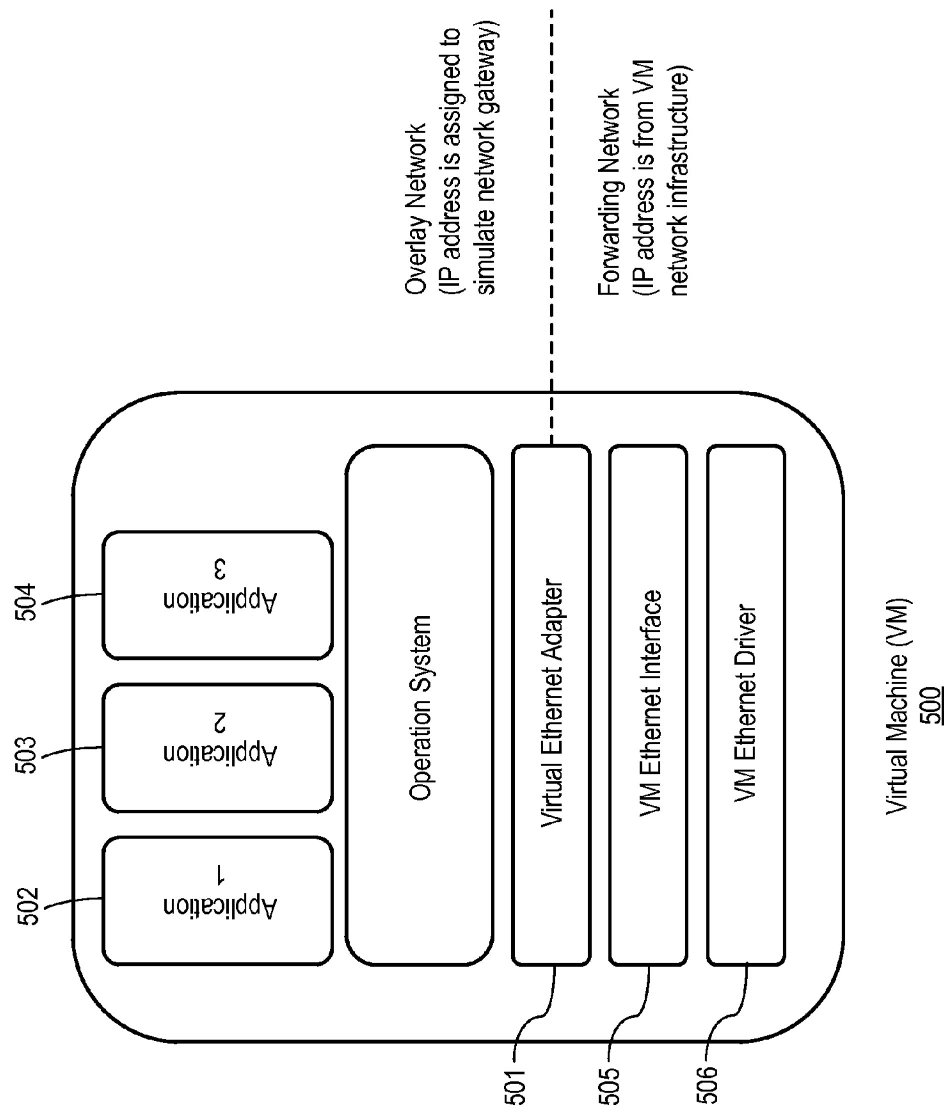

Overlay Network
(IP address is assigned to
simulate network gateway)
Forwarding Network
(IP address is from VM
network infrastructure)
504
Application
3
503
Application
2
502
Application
1
Operation System
Virtual Ethernet Adapter
VM Ethernet Interface
VM Ethernet Driver
501
505
506
Virtual Machine (VM)
500

FIG. 5

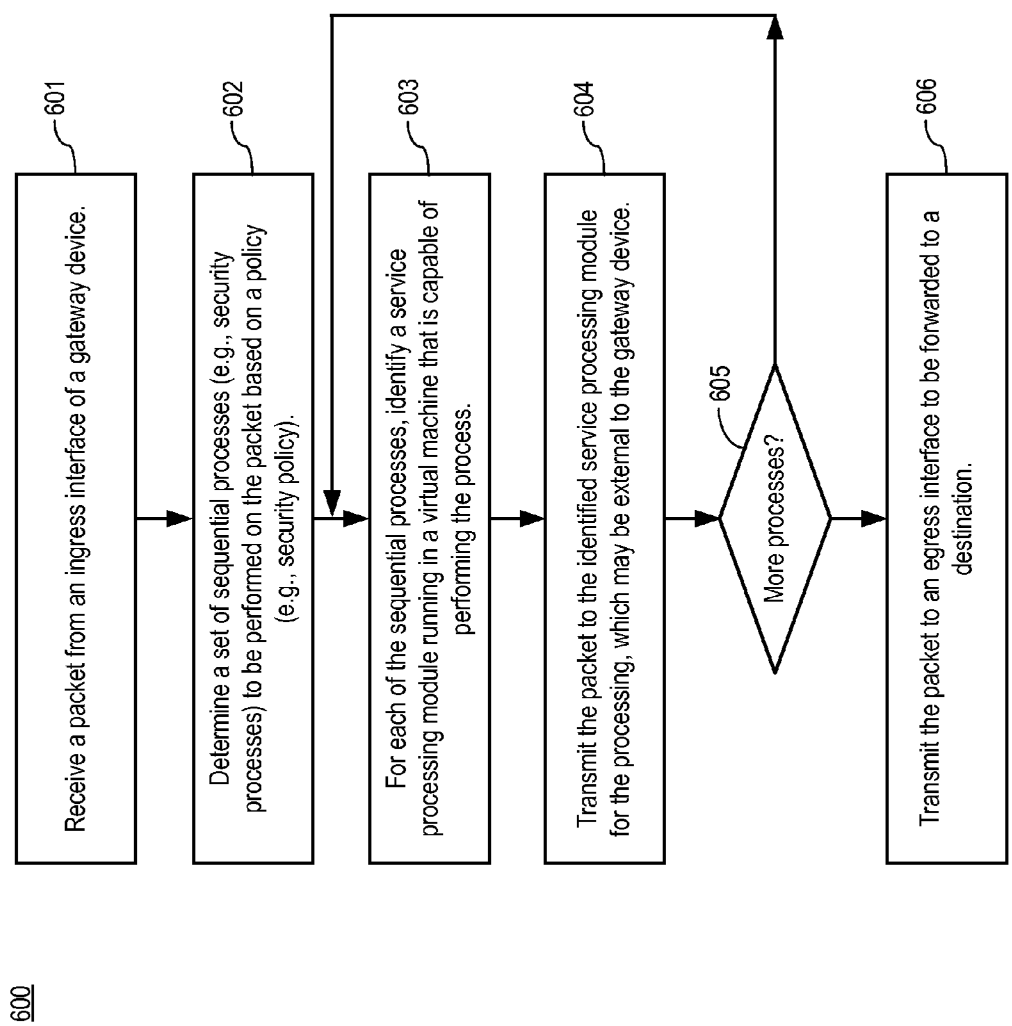

600
601
Receive a packet from an ingress interface of a gateway device.
602
Determine a set of sequential processes (e.g., security processes) to be performed on the packet based on a policy (e.g., security policy).
603
For each of the sequential processes, identify a service processing module running in a virtual machine that is capable of performing the process.
604
Transmit the packet to the identified service processing module for the processing, which may be external to the gateway device.
605
More processes?
606
Transmit the packet to an egress interface to be forwarded to a destination.

FIG. 6

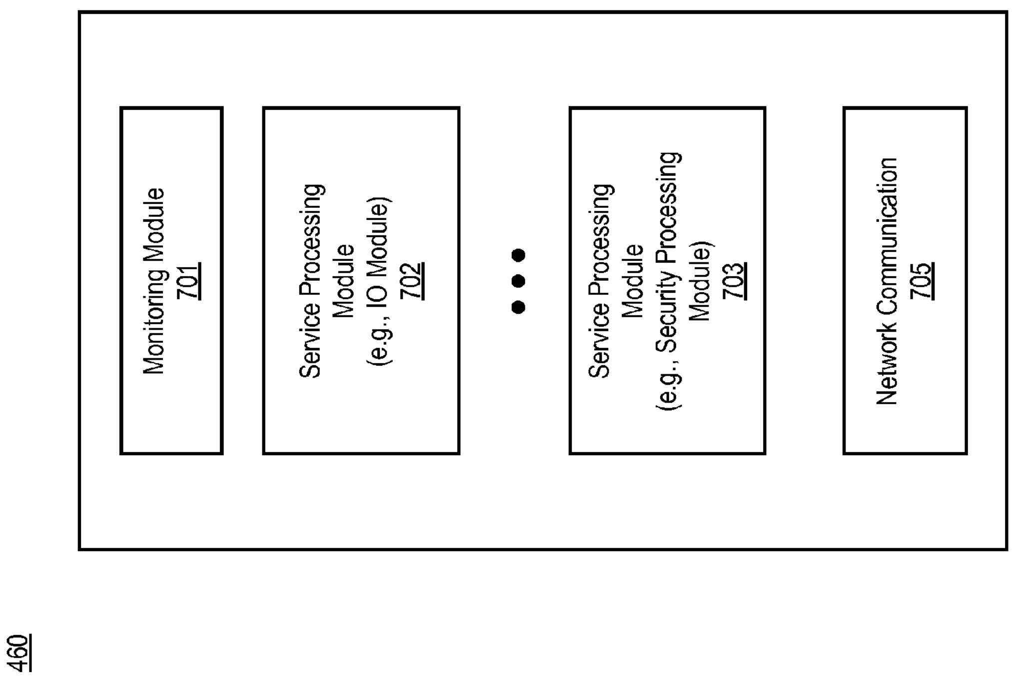

460
Monitoring Module
701
Service Processing
Module
(e.g., IO Module)
702
Service Processing
Module
(e.g., Security Processing
Module)
703
Network Communication
705

FIG. 7

800

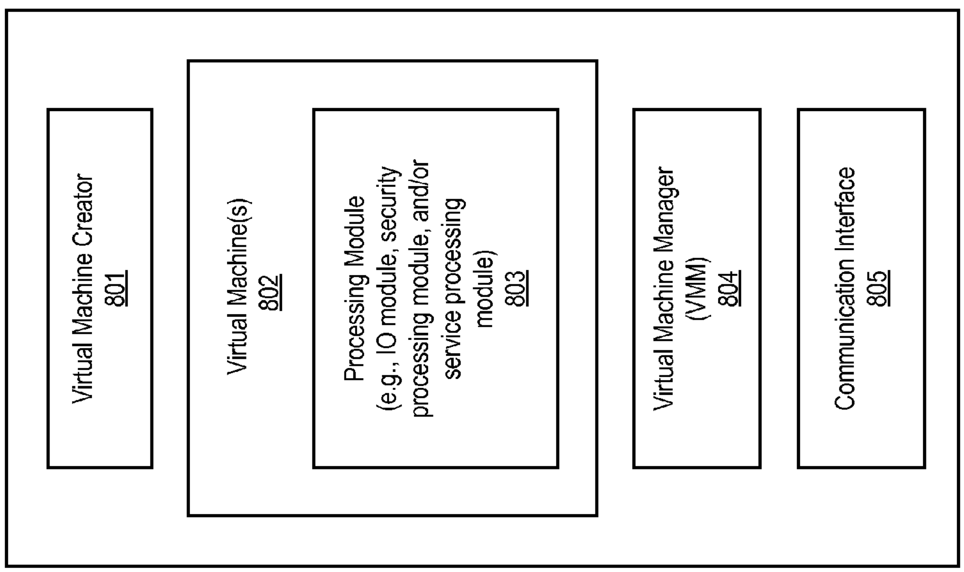

Virtual Machine Creator
801
Virtual Machine(s)
802
Processing Module
(e.g., IO module, security processing module, and/or service processing module)
803
Virtual Machine Manager
(VMM)
804
Communication Interface
805

FIG. 8

DISTRIBUTED SERVICE PROCESSING OF NETWORK GATEWAYS USING VIRTUAL MACHINES

RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Patent Application No. 61/462,980, filed Feb. 10, 2011, which is incorporated by reference in its entirety.

FIELD OF THE INVENTION

Embodiments of the present invention relate generally to network security. More particularly, embodiments of the invention relate to distributed service processing of network gateways using virtual machines.

BACKGROUND

A network gateway handles all network traffic that comes in and goes out of a network it protects. As the attacks get more sophisticated, there are more and more security and network services running on the network gateway to support the additional security functions. However, these additional services consume memory and central processing unit (CPU) resources of the gateway and limit the network throughput that the network gateway can support. Besides, if a network service must run on a particular operating system, e.g. Microsoft Server 2008, but the underlying operating system of the network gateway is different, then the gateway cannot support this network service. This limitation hinders what services the network gateway can support.

FIG. **1** is a block diagram illustrating a conventional network processing scheme in a gateway device. Referring to FIG. **1**, packets go through several network service processing stages in the network gateway, before being forwarded to next hop hosts. Typically, the packets get a sanity check (e.g., checksum, data corruption, etc.) at block **101** and then at block **102**, they are processed by a packet classifier to identify the associated connection. The packets then go through multiple network services **103**-**105** of the identified connection, before they are forwarded out of the network gateway at block **106**.

Some of the network services may need to parse the packet payload or search for patterns through the entire payload. These processes take time and memory to operate and consume valuable CPU resources otherwise could be used to process other packets. When there is a large amount of traffic and the packets go through computation-intensive services, the network gateway may slow down and cannot keep up with the traffic.

SUMMARY OF THE INVENTION

A method and apparatus is disclosed herein for distributed service processing using virtual machines. In one embodiment, the method comprises receiving a packet at an ingress interface of a gateway device interfacing a local area network (LAN) and an external network; determining a set of a plurality of processes corresponding a connections session associated with the packet based on a policy; for each of the identified processes, identifying a service processing module executed by a virtual machine that is capable of handling the identified process, and sending the packet to the identified service processing module to perform the identified process on the packet; and transmitting the packet to an egress interface of the gateway device to be forwarded to a destination.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the invention are illustrated by way of example and not limitation in the figures of the accompanying drawings in which like references indicate similar elements.

FIG. **1** is a block diagram illustrating traditional service processing in a gateway device.

FIG. **2** is a block diagram illustrating an example of a network configuration according to one embodiment of the invention.

FIG. **3** is a block diagram illustrating an example of distributed service processing according to one embodiment of the invention.

FIG. **4** is a block diagram illustrating an example of a data processing system which may be used as an embodiment of the invention.

FIG. **5** is a block diagram illustrating an architecture of a processing module according to one embodiment of the invention.

FIG. **6** is a flow diagram illustrating a method for performing distributed services according to one embodiment of the invention.

FIG. **7** illustrates a set of code (e.g., programs) and data that is stored in memory of one embodiment of a gateway according to one embodiment.

FIG. **8** illustrates a set of code (e.g., programs) and data that is stored in memory according to one embodiment.

DETAILED DESCRIPTION

Various embodiments and aspects of the inventions will be described with reference to details discussed below, and the accompanying drawings will illustrate the various embodiments. The following description and drawings are illustrative of the invention and are not to be construed as limiting the invention. Numerous specific details are described to provide a thorough understanding of various embodiments of the present invention. However, in certain instances, well-known or conventional details are not described in order to provide a concise discussion of embodiments of the present inventions.

Reference in the specification to "one embodiment" or "an embodiment" means that a particular feature, structure, or characteristic described in conjunction with the embodiment can be included in at least one embodiment of the invention. The appearances of the phrase "in one embodiment" in various places in the specification do not necessarily all refer to the same embodiment.

According to some embodiments, a new design is provided to create distributed service processing in a network gateway to support the increasing system load and to support third party services on different operating systems. An embodiment of the present invention moves at least some of these computation intensive network services into virtual machines. The virtual machines may host one or more of the network services, where the virtual machines may be hosted within the gateway or external to the gateway. Packets are forwarded to the virtual machines then forwarded back to the network gateway after the processing. The network gateway can use a load balancing mechanism to forward the packets to multiple virtual machines. Each virtual machine supports one or more network services of different connections. The load balancing to virtual machines provides a great flexibility and scalability to support a large scale of networks.

FIG. 2 is a block diagram illustrating an example of network configuration according to one embodiment of the invention. Referring to FIG. 2, gateway device 204 (also referred to as a network element, a router, a network access device, or an access point, etc.) provides an interface between network 203 and network 205. Network 203 may be an external network such as a wide area network (WAN) (e.g., Internet) while network 205 represents a local area network (LAN). Nodes 206-207 have to go through gateway device 204 in order to reach nodes 201-202, or vice versa. Any of nodes 201-202 and 206-207 may be a client device (e.g., a desktop, laptop, Smartphone, gaming device, etc.) or a server.

According to one embodiment, gateway 204 is associated with various service processing modules 208-209, each being executed within a virtual machine (VM). Each service processing module is responsible for handling one or more services. Examples of the services to be performed for gateway device 204 include, but are not limited to, network address translation (NAT), virtual private network (VPN), deep packet inspection (DPI), and/or anti-virus, etc. Some of the service processing modules are located within gateway device 204 (e.g., service modules 208) and some are located external to gateway device 204 (e.g., service modules 209 maintained by service processing node(s) 211). All of the service modules 209-210 are managed by load balancing module 210, which may be located within gateway device 204, in a public cloud associated with network 203, or in a private cloud associated with network 205. In one embodiment, load balancing module 208 and service processing modules 208-209 collectively may represent a distributed firewall of gateway 204. Further detailed information concerning a distributed firewall can be found in a co-pending U.S. patent application Ser. No. 13/363,088, entitled "Distributed Firewall Architecture using Virtual Machines," filed Jan. 31, 2012, which is incorporated by reference herein in its entirety.

A virtual machine represents a completely isolated operating environment with a dedicated set of resources associated therewith. A virtual machine may be installed or launched as a guest operating system (OS) hosted by a host OS. Typically, a host OS represents a virtual machine monitor (VMM) (also referred to as a hypervisor in one embodiment) for managing the hosted virtual machines. A guest OS may be of the same or different types with respect to the host OS. For example, a guest OS may be a Windows™ operating system and a host OS may be a LINUX operating system. In addition, the guest OSes running on a host can be of the same or different types. A virtual machine can be any type of virtual machine, such as, for example, hardware emulation, full virtualization, para-virtualization, and an operating system-level virtualization virtual machine. Different virtual machines hosted by a server may have the same or different privilege levels for accessing different resources.

FIG. 3 is a block diagram illustrating an example of a distributed service scheme according to one embodiment of the invention. Processing flow 300 may be performed by gateway device 204 of FIG. 2. Referring to FIG. 3, network service processing is shown distributed to multiple virtual machines. The network services of the gateway may be moved to external virtual machines such as services 301-302. When the packets are processed through the network service chaining in a network gateway, if the next network service is at an external virtual machine, the network gateway uses a load balancing mechanism to identify the virtual machine and then forwards the packets to the virtual machine. The load balancing algorithm may be based on round-robin, least connections, or any other well-known load balancing algorithms. The packets are sent back to the network gateway once they are processed by the virtual machine.

In one embodiment, virtual machines 301-302 can be on the same device as the network gateway, or they can reside on different devices which connect to the gateway through network connections. There are multiple possible communication protocols between the gateway and virtual machines that may be used. If the network gateway and virtual machines are in the same layer-2 network, the packet can be forwarded through a layer-2 protocol, such as, for example, the Ethernet protocol. In this case, the original IP packets are encapsulated with an Ethernet header of media access control (MAC) address of both sides. The recipient then de-encapsulates the Ethernet header and retrieves the original IP packets. The communication protocol can also be a layer-3 protocol, such as the IP protocol. The original packets are encapsulated with another IP header with the IP address of both sides. The encapsulation of the outer IP header would ensure the packets are sent and received between the virtual machine and the network gateway.

In another embodiment, the network services can be running on virtual machines or physical hosts. Running on virtual machines provides an additional benefit that additional virtual machines can be added dynamically in case of heavy traffic. Initially the network gateway may have only one virtual machine for a particular network service. When network traffic increases and the virtual machine reaches its capacity, the network gateway can utilize more virtual machines to add more system capacity. New connections are forwarded to different virtual machines for load balancing. This increases system availability and scalability.

The virtual machines 301-302 running the network services can be distributed on different networks, or at different locations, as long as the virtual machines can communicate with the network gateway. One of the examples is to put the virtual machines in a public cloud, and keep the network gateway in a data center. This provides the flexibility to add more computing resources at a lower cost, while maintaining the control of the network gateway in enterprise's premises.

FIG. 5 is a block diagram illustrating virtual machine architecture according to one embodiment of the invention. Referring FIG. 5, virtual machine 500 may be used to host any of service processing modules described above. In one embodiment, virtual machine 500 includes a virtual network adapter 501. There are at least two main functions for virtual network adapter 501. The first function is to intercept the packets coming from the network gateway, de-encapsulate the outer IP header if it uses IP protocol, then forward the packets to the applications 502-504. If there are packets being sent back to the network gateway, the virtual network adapter 501 encapsulates the destination IP address if it uses the IP protocol and then sends it to the underlying network via VM Ethernet interface 505 and VM Ethernet driver 506. This function ensures that the packets are forwarded between both sides regardless the original IP addresses of the packets.

The second function of virtual network adapter 501 is to separate the IP address of VM Ethernet interface 505 from the IP address "seen" by the applications 502-504 of virtual machine 500. As any IP address can be assigned to virtual network adapter 501, applications 502-504 on virtual machine 500 can use this IP address for application process, regardless the real IP address of VM Ethernet interface 505. The use of the separate IP address will ensure that the user-space application inserts the correct IP address in the packet payload of the application.

In further detail, according to one embodiment, virtual network adapter 501 logically creates an overlay network for virtual machine **500**. The applications **502-504** of virtual machine **500** assume the virtual IP address is the interface IP address, while the real IP address of virtual machine Ethernet interface **505** is used to transmit the data between virtual machine **500** and the network gateway. One can create as many as virtual network adapters on virtual machine **500** to simulate the target network environment, and to support a wide variety of the network topologies. The virtual machines can use any operating system, as long as the virtual network adapter driver **506** supports the operating system. Thus, the services can be supported on any operating system which may be different from the operating system the network gateway runs.

As a result, the network gateway can employ a significantly large amount of CPU and memory resources for service processing as long as it utilizes more virtual machines to support the service processing. This makes it possible that network gateway can support line-rate processing, even with most computation-intensive network services. An embodiment of the invention also allows different operating systems of the virtual machines from the one running on network gateway, which enables users to run network services on any operating systems.

In summary, an embodiment of the invention is to enable running many network services on the gateway without performance degradation. These network services may be running on an overlay network, with the freedom to have their own forwarding scenarios. Embodiments of the invention can tap the cheap resources of public cloud to run virtual machines to support a large amount of traffic without much IT investment, and provide a great scalability and system availability.

FIG. **6** is a flow diagram illustrating a method for performing distributed services according to one embodiment of the invention. Method **600** may be performed by processing logic that may include software, hardware, or a combination of both. For example, method **600** may be performed by network gateway **204** of FIG. **2**. Referring to FIG. **6**, at block **601**, a packet is received at an ingress interface of a gateway device coupling a LAN to an external network such as the Internet. At block **602**, a set of one or more sequential processes (e.g., security processes) is determined to be performed on the packet based on a policy (e.g., security policy). For each of the identified processes, at block **603**, a service processing module running within a virtual machine that is capable of performing the process is identified. For example, an existing service processing module corresponding to the process to be performed having sufficient bandwidth may be invoked. Alternatively, a new virtual machine having a new service processing module may be dynamically allocated and launched. At block **604**, the packet is transmitted to the identified service processing module for processing, where the service processing module may be located external to the gateway device and communicatively coupled to the gateway via a variety of communications protocols (e.g., Ethernet or IP protocol). The above operations involved in blocks **603-604** may be iteratively performed for each of the identified processes in the chain. Once all of the processes have been performed, at block **606**, the packet is then transmitted to an egress interface of the gateway to be forwarded to the destination.

FIG. **4** is a block diagram illustrating an example of a data processing system which may be used as an embodiment of the invention. For example, system **410** may be implemented as part of gateway device **204** or alternatively, system **410** may be implemented as part of a client or server device. In one embodiment, system **410**, which may operate as a gateway device, includes a memory, an interface to receive one or more packets from the one or more virtual machines, and one or more processors. Referring to FIG. **4**, gateway **410** includes a bus **412** to interconnect subsystems of gateway **410**, such as a processor **414**, a system memory **417** (e.g., RAM, ROM, etc.), an input/output controller **418**, an external device, such as a display screen **424** via display adapter **426**, serial ports **428** and **430**, a keyboard **432** (interfaced with a keyboard controller **433**), a storage interface **434**, a floppy disk drive **437** operative to receive a floppy disk **438**, a host bus adapter (HBA) interface card **435**A operative to connect with a Fibre Channel network **490**, a host bus adapter (HBA) interface card **435**B operative to connect to a SCSI bus **439**, and an optical disk drive **440**. Also included are a mouse **446** (or other point-and-click device, coupled to bus **412** via serial port **428**), a modem **447** (coupled to bus **412** via serial port **430**), and a network interface **448** (coupled directly to bus **412**).

Bus **412** allows data communication between central processor **414** and system memory **417**. System memory **417** (e.g., RAM) may be generally the main memory into which the operating system and application programs are loaded. The ROM or flash memory can contain, among other code, the Basic Input-Output system (BIOS) which controls basic hardware operation such as the interaction with peripheral components. Applications resident with computer system **410** are generally stored on and accessed via a computer readable medium, such as a hard disk drive (e.g., fixed disk **444**), an optical drive (e.g., optical drive **440**), a floppy disk unit **437**, or other storage medium.

Storage interface **434**, as with the other storage interfaces of computer system **410**, can connect to a standard computer readable medium for storage and/or retrieval of information, such as a fixed disk drive **444**. Fixed disk drive **444** may be a part of computer system **410** or may be separate and accessed through other interface systems.

Modem **447** may provide a direct connection to a remote server via a telephone link or to the Internet via an interne service provider (ISP). Network interface **448** may provide a direct connection to a remote server. Network interface **448** may provide a direct connection to a remote server via a direct network link to the Internet via a POP (point of presence). Network interface **448** may provide such connection using wireless techniques, including digital cellular telephone connection, a packet connection, digital satellite data connection or the like.

Many other devices or subsystems (not shown) may be connected in a similar manner (e.g., document scanners, digital cameras and so on). Conversely, all of the devices shown in FIG. **4** need not be present to practice the techniques described herein. The devices and subsystems can be interconnected in different ways from that shown in FIG. **4**. The operation of a computer system such as that shown in FIG. **4** is readily known in the art and is not discussed in detail in this application.

Code to implement the gateway operations described herein can be stored in computer-readable storage media such as one or more of system memory **417**, fixed disk **444**, optical disk **442**, or floppy disk **438**. The operating system provided on computer system **410** may be MS-DOS®, MS-WINDOWS®, OS/2®, UNIX®, Linux®, or another known operating system.

FIG. **7** illustrates a set of code (e.g., programs) and data that is stored in memory of one embodiment of a gateway, such as the gateway set forth in FIG. **4**. The gateway uses the code, in conjunction with a processor, to implement the necessary operations (e.g., logic operations) described herein. Referring to FIG. 7, the memory **460** includes a monitoring module **701** which when executed by a processor is responsible for performing traffic monitoring of traffic from the VMs as described above. Module **701** may be implemented as part of a load balancing module. Memory **460** also stores one or more service modules **702-703** which, when executed by a processor, perform any processes on the packets such as security processes. The memory **460** also includes a network communication module **704** used to perform network communication and communication with the other devices (e.g., servers, clients, etc.). For example, any of service processing modules **702-703** may be implemented as any of IO modules and security processing modules of a distributed firewall as described in the above incorporated patent application.

As described above, the servers in FIG. **1** may be implemented using a computer system. In one embodiment, one or more of the servers is implemented using a system such as depicted in FIG. **4** as well, except using different code to implement the techniques and operations performed by such servers and their VMs as described above. The code is stored in computer-readable storage medium such as system memory **417**, fixed disk **444**, optical disk **442** or floppy disk **448**.

FIG. **8** illustrates a set of code (e.g., programs) and data that is stored in one of those memories. In one embodiment of the server, such as implemented using the system shown in FIG. **4**. The server uses the code, in conjunction with the processor, to implement the necessary operations to implement the process depicted above, such as, for example, the operation set forth in FIG. **6**. Referring to FIG. **8**, the memory **800** includes virtual machine creation module **801** which when executed by a processor is responsible for creating a virtual machine on the server in a manner well-known in the art. Memory **800** also includes one or more virtual machines **802** which may be created by virtual machine creator **801**. Virtual machine **802** includes a processing module **803** executed therein, which can be one or more of an IO module, a security processing module, and/or a service processing module (e.g., NAT, VPN, DPI, anti-virus processes). Memory **800** further includes virtual machine manager (VMM) **804** responsible for managing virtual machines **802**. Memory **800** also includes communication interface module **805** used for performing communication with other devices (e.g., security gateway, servers, clients, etc.).

Some portions of the preceding detailed descriptions have been presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the ways used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the above discussion, it is appreciated that throughout the description, discussions utilizing terms such as those set forth in the claims below, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The techniques shown in the figures can be implemented using code and data stored and executed on one or more electronic devices. Such electronic devices store and communicate (internally and/or with other electronic devices over a network) code and data using computer-readable media, such as non-transitory computer-readable storage media (e.g., magnetic disks; optical disks; random access memory; read only memory; flash memory devices; phase-change memory) and transitory computer-readable transmission media (e.g., electrical, optical, acoustical or other form of propagated signals—such as carrier waves, infrared signals, digital signals).

The processes or methods depicted in the preceding figures may be performed by processing logic that comprises hardware (e.g. circuitry, dedicated logic, etc.), firmware, software (e.g., embodied on a non-transitory computer readable medium), or a combination of both. Although the processes or methods are described above in terms of some sequential operations, it should be appreciated that some of the operations described may be performed in a different order. Moreover, some operations may be performed in parallel rather than sequentially.

In the foregoing specification, embodiments of the invention have been described with reference to specific exemplary embodiments thereof. It will be evident that various modifications may be made thereto without departing from the broader spirit and scope of the invention as set forth in the following claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense.

What is claimed is:

1. A computer-implemented method, comprising:

receiving a packet at an ingress interface of a gateway device interfacing a local area network (LAN) and an external network;

determining a set of a plurality of sequential processes corresponding to a connections session to be performed on the packet based on a policy;

for each of the identified processes including a first process to be performed before a second process in the sequential processes,

identifying a service processing module that is capable of handling the identified process, wherein a first service processing module to be run in a first type of virtual machine at the gateway is identified for performing the first process, and a second service processing module to be run in a second type of virtual machine at an external network device is identified for performing the second process,

in response to a determination that the first type of virtual machine capable of executing the first service processing module is to be run by the gateway device, identifying by a load balancing module of the gateway device a first virtual machine of the first type that has sufficient bandwidth to perform the first process, and processing the packet by the first service processing module at the first virtual machine,

in response to a determination that the second type of virtual machine capable of executing the second service processing module is external to the gateway device, identifying by the load balancing module of the gateway device a second virtual machine of the second type that has sufficient bandwidth to perform the corresponding process of the second service processing module, and sending the packet to the second service processing module at the second virtual machine at the external network device to perform the second process on the packet; and

transmitting the packet to an egress interface of the gateway device to be forwarded to a destination after each of the sequential processes have been completed.

2. The method of claim 1, wherein at least one service processing module is located external to the gateway device and communicatively coupled to the gateway device.

3. The method of claim 1, wherein at least one service processing module is located in a public cloud associated with the external network.

4. The method of claim 1, wherein the plurality of sequential processes includes at least one of a network address translation (NAT) process, a virtual private network (VPN) process, a deep packet inspection (DPI) process, or an anti-virus process.

5. The method of claim 1, further comprising determining by the load balancing module whether an existing service module has sufficient bandwidth to handle the corresponding process.

6. The method of claim 5, further comprising allocating and launching by the load balancing module the first service processing module if there is no existing service processing module that has sufficient bandwidth to perform the corresponding process of the packet.

7. A non-transitory computer-readable storage medium having instructions stored therein, which when executed by a computer, cause the computer to perform a method, the method comprising:

receiving a packet at an ingress interface of a gateway device interfacing a local area network (LAN) and an external network;

determining a set of a plurality of processes corresponding a connections session associated with the packet based on a policy;

for each of the identified processes, including a first process to be performed before a second process in the sequential processes,

identifying a service processing module that is capable of handling the identified process, wherein a first service processing module to be run in a first type of virtual machine at the gateway is identified for performing the first process, and a second service processing module to be run in a second type of virtual machine at an external network device is identified for performing the second process,

in response to a determination that the first type of virtual machine capable of executing the first service processing module is to be run by the gateway device, identifying by a load balancing module of the gateway device a first virtual machine of the first type that has sufficient bandwidth to perform the first process, and processing the packet by the first service processing module at the first virtual machine,

in response to a determination that the second type of virtual machine capable of executing the second service processing module is external to the gateway device, identifying by the load balancing module of the gateway device a second virtual machine of the second type that has sufficient bandwidth to perform the corresponding process of the second service processing module, and sending the packet to the second service processing module at the second virtual machine at the external network device to perform the second process on the packet; and

transmitting the packet to an egress interface of the gateway device to be forwarded to a destination after each of the sequential processes have been completed.

8. The non-transitory computer-readable storage medium of claim 7, wherein at least one service processing module is located external to the gateway device and communicatively coupled to the gateway device.

9. The non-transitory computer-readable storage medium of claim 7, wherein at least one service processing module is located in a public cloud associated with the external network.

10. The non-transitory computer-readable storage medium of claim 7, wherein the plurality of sequential processes includes at least one of a network address translation (NAT) process, a virtual private network (VPN) process, a deep packet inspection (DPI) process, and an anti-virus process.

11. The non-transitory computer-readable storage medium of claim 7, wherein the method further comprises determining by the load balancing module whether an existing service module has sufficient bandwidth to handle the corresponding process.

12. The non-transitory computer-readable storage medium of claim 11, wherein the method further comprises allocating and launching by the load balancing module the first service processing module if there is no existing service processing module that has sufficient bandwidth to perform the corresponding process of the packet.

13. A gateway device, comprising:

an ingress interface;

an egress interface; and

a processor coupled to the ingress and egress interfaces that executes a load balancing module to

receive a packet from the ingress interface,

determine a set of a plurality of sequential processes corresponding a connections session to be performed on the packet based on a policy,

for each of the identified processes including a first process to be performed before a second process in the sequential processes, perform the following operations:

identifying a service processing module that is capable of handling the identified process, wherein a first service processing module to be run in a first type of virtual machine at the gateway is identified for performing the first process, and a second service processing module to be run in a second type of virtual machine at an external network device is identified for performing the second process,

in response to a determination that the first type of virtual machine capable of executing the first service processing module is to be run by the gateway device, identifying by a load balancing module of the gateway device a first virtual machine of the first type that has sufficient bandwidth to perform the first process, and processing the packet by the first service processing module at the first virtual machine,

in response to a determination that the second type of virtual machine capable of executing the second service processing module is external to the gateway device, identifying by the load balancing module of the gateway device a second virtual machine of the second type that has sufficient bandwidth to perform the corresponding process of the second service processing module, and sending the packet to the second service processing module at the second virtual machine at the external network device to perform the second process on the packet, and

transmit the packet to the egress interface of the gateway device to be forwarded to a destination after each of the sequential processes have been completed.

14. The gateway device of claim **13**, wherein at least one service processing module is located external to the gateway device and communicatively coupled to the gateway device.

15. The gateway device of claim **13**, wherein at least one service processing module is located in a public cloud associated with the external network.

16. The gateway device of claim **13**, wherein the plurality of sequential processes includes at least one of a network address translation (NAT) process, a virtual private network (VPN) process, a deep packet inspection (DPI) process, and an anti-virus process.

17. The gateway device of claim **13**, wherein the processor executes the load balancing module to determine whether an existing service module has sufficient bandwidth to handle the corresponding process.

18. The gateway device of claim **17**, wherein the processor executes the load balancing module to allocate and launch the first service processing module if there is no existing service processing module that has sufficient bandwidth to perform the corresponding process of the packet.

* * * * *